Report Number: WUCS-98-27

1998-01-01

Agnostic Learning of Geometric Patterns

Sally A. Goldman, Stephen S. Kwek, and Stephen D. Scott

Goldberg, Goldman, and Scott demonstrated how the problem of recognizing a landmark from a one-dimensional visual image can be mapped to that of learning a one-dimensional geometric pattern and gave a PAC algorithm to learn that class. In this paper, we present an efficient on-line agnostic learning algorithm for learning the class of constant-dimension geometric patterns. Our algorithm can tolerate both classification and attribute noise. By working in higher dimensional spaces we can represent more features from the visual image in the geometric pattern. Our mapping of the data to a geometric pattern, and hence our learning algorithm, is...
Read complete abstract on page 2.

Agnostic Learning of Geometric Patterns

Sally A. Goldman, Stephen S. Kwek, and Stephen D. Scott

Complete Abstract:

Goldberg, Goldman, and Scott demonstrated how the problem of recognizing a landmark from a one-dimensional visual image can be mapped to that of learning a one-dimensional geometric pattern and gave a PAC algorithm to learn that class. In this paper, we present an efficient on-line agnostic learning algorithm for learning the class of constant-dimension geometric patterns. Our algorithm can tolerate both classification and attribute noise. By working in higher dimensional spaces we can represent more features from the visual image in the geometric pattern. Our mapping of the data to a geometric pattern, and hence our learning algorithm, is applicable to any data representable as a constant-dimensional array of values, e.g. sonar data, temporal difference information, or amplitudes of a waveform. To our knowledge, these classes of patterns are more complex than any class of geometric patterns previously studied. Also, our results are easily adapted to learn the union of fixed-dimensional boxes from multiple-instance examples. Finally, our algorithms are tolerant of concept shift.

Agnostic Learning of Geometric Patterns

Sally A. Goldman, Stephen S. Kwek and
Stephen D. Scott

WUCS-98-27

December 1998

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

Agnostic Learning of Geometric Patterns*

Sally A. Goldman†
Dept. of Computer Science
Washington University
St. Louis, MO 63130-4899
sg@cs.wustl.edu

Stephen S. Kwek†
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-1035
kwek@eecs.wsu.edu

Stephen D. Scott†
Dept. of Computer Science and Engineering
University of Nebraska
Lincoln, NE 68588-0115
sscott@cse.unl.edu

WUCS-98-27

December 1998

Abstract

Goldberg, Goldman, and Scott demonstrated how the problem of recognizing a landmark from a one-dimensional visual image can be mapped to that of learning a one-dimensional geometric pattern and gave a PAC algorithm to learn that class. In this paper, we present an efficient on-line agnostic learning algorithm for learning the class of constant-dimension geometric patterns. Our algorithm can tolerate both classification and attribute noise. By working in higher dimensional spaces we can

*An earlier version appears in *Proceedings of the Tenth Annual ACM Conference on Computational Learning Theory*, 1997

†Supported in part by NSF NYI Grant CCR-9357707 with matching funds provided by Xerox PARC and WUTA.

represent more features from the visual image in the geometric pattern. Our mapping of the data to a geometric pattern, and our hence our learning algorithm, is applicable to any data representable as a constant-dimensional array of values, e.g. sonar data, temporal difference information, or amplitudes of a waveform. To our knowledge, these classes of patterns are more complex than any class of geometric patterns previously studied. Also, our results are easily adapted to learn the union of fixed-dimensional boxes from multiple-instance examples. Finally, our algorithms are tolerant of concept shift.

Keywords: Learning geometric patterns, agnostic learning, landmark matching problem, Winnow, virtual weights, multiple-instance examples, concept shift, robot navigation

1 Introduction

Consider a robot designed to navigate through a large-scaled environment[1]. Suppose a set of key "landmarks" have already been selected (by another component of the navigation system). It is crucial that the robot be able to recognize whether or not it is in the vicinity of a given landmark from data taken at the robot's current location. We refer to this problem as the *landmark matching problem.* The landmark matching algorithm should run in real-time and be noise-tolerant.

A common approach to designing landmark matching algorithms uses a pattern matching approach to match the visual image (or whatever form of data is available) to the data taken at landmark position L. The matching algorithm must determine if the robot is near L (i.e. in a small circle centered around L). Because the visual image may change significantly as small movements around L are made, the pattern matching approach encounters difficulties. Goldberg, Goldman and Scott [11, 13] proposed using a learning algorithm to construct an accurate hypothesis for performing landmark matching. They obtained their training data by converting the visual data into one-dimensional geometric patterns. Then by applying their algorithm, giving it a set of positive examples (i.e. patterns obtained from locations in the vicinity of the landmark) and a set of negative examples (i.e. patterns obtained from locations not in the vicinity of the landmark), their algorithm constructs a hypothesis to accurately predict if the robot is near the given landmark.

While the basic approach suggested by Goldberg, Goldman and Scott of using learning versus pattern matching for the landmark matching problem can be applied to a wide range of data, the rest of their work was specific to the data from an imaging system that generates a one-dimensional array of light intensities (called a *signature*) taken at eye level [17, 22, 29, 33]. The motivation for using one-dimensional data is to reduce the processing time. For some settings, such as an office environment, it seems feasible that the signature taken at eye level is sufficient. On the other hand, if one wants to design a landmark matching data for a Mars rover, such an approach would not work. In this work, we extend the basic approach of Goldberg, Goldman and Scott so that different types (and dimensionality) of data can be used.

[1]By a large-scaled environment we mean that not all landmarks are visible from all locations in the environment.

So that we can relate our work to that of Goldberg, Goldman, and Scott, we briefly describe the class of one-dimensional geometric patterns. In a (discretized and bounded) one-dimensional geometric pattern, the "target" pattern is a configuration (collection) of up to k points from $\{1, \ldots, s\}$. Each example (instance) is a configuration of up to n points from $\{1, \ldots, s\}$, where it is labeled according to whether or not it visually resembles the target pattern based on the *Hausdorff metric* (for example, see Gruber [14]). Goldberg, Goldman and Scott [11] gave an Occam-based PAC algorithm for learning the class of one-dimensional geometric patterns from the continuous domain. Following that work, Goldman and Scott [13] gave a statistical query algorithm (and hence a noise-tolerant PAC algorithm) for the same class.

One contribution of this paper is an on-line agnostic learning algorithm (that tolerates classification and attribute noise) for learning the class of discretized one-dimensional geometric patterns. (The class we study is a slight generalization of a discretized and bounded version of the class Goldberg, Goldman and Scott studied.) We obtain our algorithm by reducing the problem to that of learning a disjunction of a large (exponential) set of variables. We then apply Winnow [23] to obtain our agnostic learning algorithm, and the virtual weight technique of Maass and Warmuth [28] to make our algorithm efficient.

In some experimental work performed by Goldman and Scott [13] it was found that in moving from the processed one-dimensional visual image to a one-dimensional pattern, too much key information was lost. We define a class of two-dimensional geometric patterns for which the important features from the visual image are incorporated in the two-dimensional pattern. We then show how to extend our agnostic learning algorithm to this class of two-dimensional patterns. Furthermore, we use our construction to obtain an efficient agnostic learning algorithm for the class of d-dimensional geometric patterns[2] for d any constant where each example is a collection of up to n points from $\{1, \ldots, s\}^d$. For any sequence of trials, the mistake bound for our algorithm is polynomial in k, n, $\log(s)$ and M_{opt}, the number of mistakes made by the best pattern. So, for example, we can apply this algorithm to the problem of recognizing a landmark from two-dimensional data (where each data item is any number such as a light intensity, sonar data, temporal difference information, or amplitudes of a waveform) by mapping it into a three (or higher)-dimensional geometric pattern. To our knowledge, these classes of patterns are more complex than any class of geometric patterns previously studied. Finally, our algorithms are tolerant of concept shift.

There is also a relationship between this work and the task of learning from multiple-instance examples [10]. In the multiple-instance learning model, the target concept is a boolean function, each example is a collection of instances, and the example (collection) is classified as positive iff at least one of its elements is mapped to positive by the target concept. Long and Tan [27] described an efficient PAC algorithm for learning a single axis-parallel box in $\mathbf{Q}^d$ from multiple-instance examples under a product distribution where $\mathbf{Q}$ denotes the set of rationals and d need not be constant. Auer et al. [4] gave an efficient PAC algorithm for learning a single axis-parallel box in $\Re^d$ from multiple-instance examples if each instance is drawn independently from an arbitrary distribution over $\Re^d$. This algorithm also

[2]Note that while we reduce our problem to learning a disjunction of boxes, the class itself is more complex than of unions of boxes. First, each example is a multiple-instance example since it contains n points. Also, the disjunction we form defines the complement of the target concept and is composed of some boxes that are *not* those defining the target concept. (See Section 4 for further discussion about this topic for $d = 1$.)

runs in time that is polynomial in d. Later, Auer [3] modified that algorithm (making it more practical) and gave an empirical analysis of the modified algorithm. In the above papers, each example is classified as positive if at least one of its points is inside the target box. Our algorithm for learning constant-dimensional patterns can be viewed as learning a union of axis-parallel boxes from a constant-dimensional discretized and finite space where a more complex rule is used for specifying when an example is classified as positive. Namely, a multiple-instance example is positive iff (1) each point is classified as positive by some box and (2) every box contains at least one point. Furthermore, our algorithm is easily adapted to use the rule that an example is positive if at least one of its points is inside some target box (or other variations).

This paper is organized as follows. In the next section we discuss the landmark matching problem in more depth. Then, in Section 3, we review the background material for this paper. In Section 4 we formally define the concept class of d-dimensional geometric patterns. In Section 5 we present our generic algorithm for learning d-dimensional patterns for constant d and give specific mistake bounds for learning one-dimensional patterns. Section 6 formalizes our new mapping from signatures to two-dimensional patterns and gives the mistake bound for the two-dimensional pattern learner. Section 7 gives other extensions of our algorithm, and Section 8 describes how our algorithm can be used to learn the class of unions of axis-parallel boxes in fixed dimension when *multi-instance examples* are provided (i.e. an example is positive if *any* of its points lies in the target box). This class generalizes the class of single axis-parallel boxes with multi-instance examples, which has potential applications in the area of drug discovery [10]. We conclude in Section 9.

2 The Landmark Matching Problem

In this section we explore in further depth how this work might apply to the landmark matching problem (or other signal processing problems). We make no assumptions about the imaging system except that it provides a constant-dimensional array of reals (that in turn might be light intensities, sonar data, ...) A common approach to designing landmark matching algorithms uses pattern matching by trying to match the current image to the image taken at landmark position L with some fixed orientation. If one's goal is to determine if the robot is standing exactly at position L with that same orientation, then the pattern matching approach can be implemented to work well. However, in reality, the matching algorithm must determine if the robot is in the vicinity of L (i.e. in a circle centered around L with an orientation that is similar to what is expected). Because the image may change significantly as small movements around L and rotations are made, the pattern matching approach encounters difficulties.

Rather than using a pattern matching approach to match the image from the current location with the image of the landmark, we instead propose using a learning algorithm to construct a good hypothesis for performing landmark matching. Intuitively, the learning algorithm is being used to combine a set of positive examples to create a hypothesis that will make good predictions. We obtain the instances by converting the image into a geometric pattern by placing points where there are significant changes. For example, a 1-d array of light intensities could be mapped to a two-dimensional pattern where the second dimensions

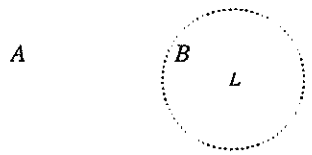

Figure 1: An example to help illustrate a problem that must be overcome. Suppose A and B are the locations of the bases of an arch and L is a landmark location. The visual images obtained within the dashed circle change dramatically.

corresponds to the value of the derivative of the intensity array. Then by applying our algorithm, giving it a set of positive examples (i.e. patterns[3] obtained from locations in the vicinity of the landmark with slight variations in orientation) and a set of negative examples (i.e. patterns obtained from locations not in the vicinity of the landmark), we construct a hypothesis that can accurately predict whether or not the robot is near the given landmark with the expected orientation, assuming that the positive and negative examples are sufficiently distinct.

A natural question raised is why there is a need for learning here. To answer this question, we briefly examine some problems that would occur if a single pattern taken at the landmark location was used as a hypothesis. (The same problems cause difficulties when using a pattern matching approach.) Suppose a landmark location was selected at the position L shown in Figure 1 where A and B are the legs of the Gateway Arch in downtown St. Louis. (For simplicity, suppose that nothing besides the arch is in the robot's visual image.) Further, suppose we want the landmark matching system to indicate that the robot is at landmark position L exactly when it is in the dashed circle with and orientation between 355° and 5°, where 0° is due north. Clearly in this example, the robot's visual image changes dramatically as the robot moves within this circle and rotates. Thus simply using as a hypothesis the single pattern obtained from the visual image obtained from the landmark location and oriented at 0° would not yield good predictions as to whether or not the robot is "near" the landmark location. Additionally, if we instead use an instance-based approach, where we attempt to match the robot's visual image to one of several images (shot within the circle) stored in the robot's memory, then we might violate some on-line time and space constraints.

In the learning-based approach we propose here, the navigation system (when selecting L as a landmark) would collect images from locations evenly spaced throughout the dashed circle with varying orientations. Then by using these images as positive examples (and images taken at random locations not in the circle as negative examples), we can apply our learning algorithm to combine these images to obtain a hypothesis to predict if the robot is near L. A valuable feature of the learning algorithm is the generality of its hypothesis class. The concepts (defined precisely in Section 4) are *approximations* to sets of patterns visible within limited regions, and this more general hypothesis class may allow a better fit to the diverse set of positive examples than the simple concept of a single pattern.

Also, when really applying an algorithm for learning geometric patterns to the landmark

[3]We assume that the portion of the complete navigation system that selects the landmarks will gather a set of images near the landmark to be used as the positive examples for training.

matching problem, one must handle noisy data. Because of the noise inherent in the data, the problems illustrated above with simply using as a hypothesis the single (noisy) pattern obtained from the visual image at the landmark location would be exacerbated. Thus in this paper we present algorithms that can provably tolerate certain types of noise in the data. Our belief is that these algorithms will be robust against many types of noise, including those for which they have no known theoretical guarantees.

3 The On-Line, Agnostic Learning Model and Winnow

In this paper we consider the on-line (or mistake-bound) learning model [1, 23] as applied to concept learning (i.e. the classification of each example is 1 or 0). The learning proceeds in *trials*, where in trial t an example X_t is presented to the learner, and in polynomial time the learner must produce a prediction $\hat{y}_t$ as to the classification of X_t. Then the learner receives the desired output y_t and incurs a *loss* $L(y_t, \hat{y}_t)$ for some loss function. Since we focus on concept learning, we use the *loss* function: $L(y_t, \hat{y}_t)$ is 1 if $y_t \neq \hat{y}_t$, and 0 otherwise. The performance of the on-line learner is measured by the total loss over all trials, which is equivalent to the number of prediction mistakes made when using the discrete loss function. Our on-line learning algorithms are *agnostic* [15, 19] in the sense that they make no assumptions whatsoever about the target concept to be learned. Instead, we compare their performance with the performance of the *best hypothesis* selected from a comparison or "touchstone" class. For a sequence of trials, the best hypothesis from the touchstone class is the one that makes the minimum number of mistakes. We say that an algorithm has polynomial complexity if its mistake bound and time complexity are polynomial in the number of bits required to specify an example and the number of bits needed to encode the best hypothesis.

An important result in this model is Littlestone's on-line noise-tolerant algorithm Winnow [23] for learning K-disjunctions of boolean attributes when there is a large number N of total attributes and $N \gg K$. (The $N - K$ attributes not in the target K-disjunction are considered *irrelevant.*) Winnow makes predictions based on a linear threshold function

$$\hat{y}_t = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} w_i x_i \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where w_i is the weight associated with the boolean attribute x_i. If the prediction is wrong then the weights are updated as follows. On a false negative prediction, for each attribute x_i that is 1, Winnow *promotes* the weight w_i by multiplying w_i by some constant update factor $\alpha > 1$. On a false positive prediction, for each attribute x_i that is 1, Winnow *demotes* the weight w_i by dividing it by α. Winnow is similar to the classical Perceptron algorithm [31], except that the Perceptron algorithm updates its weights *additively* while Winnow uses *multiplicative* weight updates. Another major difference between these algorithms is that Winnow's mistake bound is logarithmic in N whereas the Perceptron algorithm's mistake bound can be linear in N in the worst case [20].

Recently Auer and Warmuth [5], in generalizing the work of Littlestone [25], showed that Winnow makes at most $O(A + K \log N)$ mistakes on any sequence of trials where the target K-disjunction makes at most A attribute errors. The number of attribute errors of a labeled

example $\langle X_t, y_t \rangle$ with respect to the target disjunction is the minimum number of attributes (bits) of X_t that have to be changed so that the classification of the resulting example by the target is consistent with y_t. In the agnostic model, whenever the best hypothesis makes a prediction mistake, we only need to change at most K attributes of the example so that the classification is consistent. Thus we have the following interpretation of the mistake bound in the presence of attribute errors[4].

Theorem 1 *[5] Suppose in a sequence of trials for on-line learning an unknown boolean concept defined by $\leq K$ of N possible attributes, the best K-disjunction makes M_{opt} mistakes (classification errors). Then Winnow, running with $\alpha = 1.75$, each initial weight $= 1/N$, and $\theta = (\alpha \ln \alpha)/(\alpha^2 - 1)$, makes at most*

$$2.75KM_{opt} + 4.92K(\ln N - 1) + 4.92$$

mistakes.

Auer and Warmuth [5] also offer a version of Winnow that tolerates concept shift (i.e. the target disjunction may change completely in time). When a weight is sufficiently small, they do not demote it any further. Specifically, no weight is allowed to fall below β/N for some $\beta \geq 0$. Let z_t be the total number of additions and deletions of literals to the best K-disjunction in trial t. Then the total number of shifts in T trials is $Z = \sum_{t=1}^{T} z_t$. Again, since a prediction mistake by the optimal K-disjunction corresponds to at most K attribute errors, we get the following mistake bound in the presence of shifting concepts.

Theorem 2 *[5] Suppose in a sequence of trials for on-line learning a sequence of unknown boolean concepts, each defined by $\leq K$ of N possible attributes, the best sequence of K-disjunctions makes M_{opt} mistakes (classification errors) and the sequence of disjunctions includes Z total shifts. Then Winnow, running with $\alpha = 1.32$, $\beta = 0.0269$, each initial weight $= \beta/N$, and $\theta = (\alpha \ln \alpha + (\alpha - 1)\beta)/(\alpha^2 - 1)$, makes at most*

$$2.4KM_{opt} + 4.32Z(\ln N + 3.89) + 4.32 \min\{N, Z\}(\ln N + 1.34) + 0.232$$

mistakes.

It can be shown that the bounds of Theorems 1 and 2 are optimal up to constants if no information other than N is known [5]. Better bounds can be achieved if we tune Winnow using knowledge of K, Z, and A. One method of tuning Winnow is to apply the weighted majority algorithm, as demonstrated by Littlestone and Warmuth [26].

One more issue we must contend with is that a direct implementation of Winnow in our setting requires that the number of attributes (and thus the computation time) be exponential in the number of bits required to represent an instance and a concept. We circumvent this problem by applying the virtual weight technique of Maass and Warmuth [28] to implicitly maintain the weights. The basic idea is to simulate Winnow by grouping

[4]Note that we can interpret attribute errors as noise in the attributes or classification noise instead of an agnostic result. We will present our results in terms of agnosticism since they seem as appropriate to our application and as easily comprehensible as the other interpretations.

concepts that "behave alike" into blocks. For each block only one weight has to be computed and we construct the blocks so that the number of concepts combined in each block as well as the weight for the block can be efficiently computed. While the number of blocks increases as new counterexamples are received, the total number of blocks we create is polynomial in the number of mistakes. Using the virtual weights technique will allow our time complexity to be polynomial in the size of the examples and the size of the target concept.

It is also well known that learning algorithms in the mistake-bound model can be mapped to PAC algorithms [1, 24]. A PAC (probably approximately correct) algorithm [34, 35] is an off-line algorithm that draws examples randomly according to an arbitrary, unknown probability distribution and with probability $\geq 1-\delta$ outputs a hypothesis with error $\leq \epsilon$, where the parameters ϵ and δ are given as inputs to the algorithm. We can convert our mistake-bound algorithm to a PAC algorithm as follows. For each trial, we draw $q_i = \lceil (1/\epsilon)(\ln(1/\delta) + i \ln 2) \rceil$ examples randomly according to $\mathcal{D}$ and check if our current hypothesis (the setting of the weights in Winnow) is consistent with the sample. If it is, then we halt. Otherwise we take one of the examples our hypothesis misclassifies and use it to update the weights. We then move on to the next trial. For a mistake bound of M, the total sample complexity is

$$\begin{aligned} \sum_{i=1}^{M} q_i &= \frac{1}{\epsilon} \sum_{i=1}^{M} \left(\ln \frac{1}{\delta} + i \ln 2 \right) \\ &= \frac{M}{\epsilon} \left(\ln \frac{1}{\delta} + \frac{(M+1)\ln 2}{2} \right) = O\left(\frac{1}{\epsilon} \left(M \ln \frac{1}{\delta} + M^2 \right) \right) \end{aligned}$$

and the time complexity is $O(N)$ times the sample complexity since it takes $O(N)$ time to evaluate the hypothesis on each example. Note that the mistake bound M need not be known in this mapping. This gives us the following theorem.

Theorem 3 *[1, 23] Winnow with an unknown mistake bound M can be converted to a PAC algorithm with sample complexity*

$$\frac{M}{\epsilon} \left(\ln \frac{1}{\delta} + \frac{(M+1)\ln 2}{2} \right).$$

The time complexity is $O(N)$ times the sample complexity.

A better bound can be attained if M is known [24]. Littlestone showed that it is possible to convert a mistake-bounded algorithm to a PAC algorithm with sample complexity $O\left(\frac{1}{\epsilon}\left(\ln \frac{1}{\delta} + M\right)\right)$ by drawing a sample and then running the on-line algorithm on the sample, saving all the hypotheses created by the on-line algorithm. Then hypothesis testing [16] is used to select the best hypothesis by evaluating them all on a new sample. We repeat his main result here.

Theorem 4 *[24] Winnow with a known mistake bound[5] M can be converted to a PAC algorithm with sample complexity*

$$\frac{1}{\epsilon} \left(48 \ln \frac{2}{\delta} + 4M + 32 \ln(M+2) - 2 \right) = O\left(\frac{1}{\epsilon} \left(\log \frac{1}{\delta} + M \right) \right).$$

[5]Technically, all we require is that M is a known upper bound on the mistake bound.

The time complexity is

$$O\left(\left(\frac{1}{\epsilon}\log\frac{M}{\delta}\right)MN\right).$$

The time complexity comes from the fact that we draw $O\left(\frac{1}{\epsilon}\log\frac{M}{\delta}\right)$ examples for hypothesis testing, we test $\leq M$ hypotheses, and each test takes $O(N)$ time.

4 The Class of One-Dimensional Geometric Patterns

We now define the concept class of one-dimensional geometric patterns for the discretized domain $\{1,\ldots,s\}$. The instance space $\mathcal{X}_n$ consists of all configurations of at most n points[6] from $\{1,\ldots,s\}$. A concept is the set of all configurations from $\mathcal{X}_n$ within some distance[7] γ under the Hausdorff metric of some "ideal" configuration of at most k points. The Hausdorff distance between configurations P and Q, denoted $\mathrm{HD}(P,Q)$, is

$$\max\left\{\max_{p\in P}\left\{\min_{q\in Q}\{dist(p,q)\}\right\},\max_{q\in Q}\left\{\min_{p\in P}\{dist(p,q)\}\right\}\right\}$$

where $dist(p,q)$ is the distance between p and q. In words, if each point in P reports the distance to its nearest neighbor in Q and each point in Q reports the distance to its nearest neighbor in P, then the Hausdorff distance is the maximum of these distances. Thus $\mathrm{HD}(P,Q)\leq\gamma$ if every point in P is within distance γ of some point in Q and every point in Q is within distance γ of some point in P. For $P\in\mathcal{X}_k$, we define the concept C_P that corresponds to P by $C_P=\{X\in\mathcal{X}_n \mid \mathrm{HD}(P,X)\leq\gamma\}$ (for ease of exposition, in this paper we assume $\gamma=1$). Figure 2 illustrates an example of such a concept. Thus one can view each concept as a sphere of unit radius in a metric space where P defines the center of the sphere. For any $X\in\mathcal{X}_n$ such that $X\in C_P$, we say that X is a *positive example* of C_P. Likewise, if $X\notin C_P$, we say that X is a *negative example* of C_P. Furthermore, all configurations of points that resemble the given configuration P are contained within this sphere. Finally, the concept class $\mathcal{C}_{k,n}$ that we study is defined as follows.

Definition 1

$$\mathcal{C}_{k,n}\doteq\{C_P \mid P \textit{ is a configuration of} \leq k \textit{ points from } \{1,\ldots,s\}\}.$$

It may be the case that $n\gg k$. For example, the learner may be asked to predict if a configuration of 100 points is contained within a sphere defined by 3 points. This consideration is, in some sense, analogous to the notion of irrelevant attributes studied in the boolean domain (Section 3). Namely, given any positive (respectively, negative) example from $\mathcal{X}_n$, there exists a subset of k of the n points in that example such that the configuration of these k points is also a positive (respectively, negative) example. However, observe that unlike the boolean domain, there is no fixed set of points of an instance that are "relevant".

[6]Note that throughout this paper, the word "point" will refer to a single point, and we shall use the term "a configuration of points" when speaking of an example.

[7]When applying our learning algorithm to the landmark matching problem, γ can be different for each landmark.

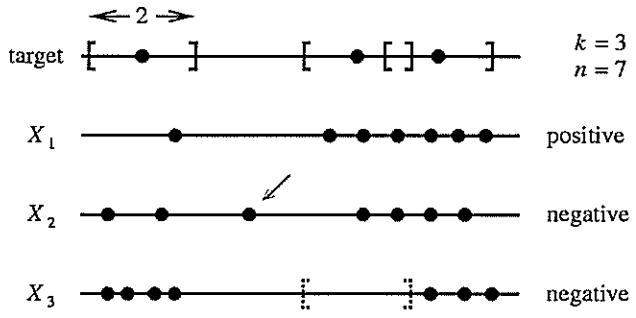

Figure 2: This figure illustrates an example concept from $\mathcal{C}_{3,7}$. The top line shows the target pattern. Around each target point we show an interval that covers all points within unit distance from that point. Every positive example must have every point within one of the above intervals *and* no interval can be empty (e.g. see X_1 above). For an example to be negative, there must be a point in it that is not within unit distance of any target point (e.g. X_2) and/or there are no points in the example near some target point (e.g. X_3).

Thus if an arbitrary point is removed from an instance it can no longer be determined if that instance was positive or negative before the point was removed.

At first glance, there may appear to be some similarities between $\mathcal{C}_{k,n}$ and the class of the union of at most k intervals over the real line. However, the class of one-dimensional geometric patterns is really quite different (and significantly more complex) than the class of unions of intervals on the real line. One major difference is that for the union of intervals, each instance is a single point on the real line, whereas for $\mathcal{C}_{k,n}$ each instance is a set of n points on the real line. Thus the notion of being able to independently vary the concept complexity and instance complexity does not exist for the class of unions of intervals. Furthermore, observe that for $\mathcal{C}_{k,n}$ each instance (configuration of n points) is an element of a metric space, which has a measure of distance defined between any pair of instances. However, with the class of unions of intervals there is no notion of a distance between instances. Finally, for the class of unions of intervals, an instance is a positive example simply when the single point provided is contained within one of the k intervals. For $\mathcal{C}_{k,n}$ an instance is positive if and only if it satisfies the following two conditions.

1. Each of the n points in the instance is contained within one of the k width-2 intervals defined by the k target points.

2. There is at least one of the n points in the instance contained within the width-2 interval defined by each of the k target points.

Further we note that Goldberg [12] has shown that it is NP-complete to find a sphere in the given metric space (i.e. one-dimensional patterns of points on the line under the Hausdorff metric) consistent with a given set of positive and negative examples of an unknown sphere in the given metric space. In other words, given a set $\mathcal{S}$ of examples labeled according to some one-dimensional geometric pattern of k points, it is NP-complete to find some one-dimensional geometric pattern (of *any* number of points) that correctly classifies all examples

in $\mathcal{S}$. So for this concept class it is NP-hard to solve the *consistent hypothesis problem*, and by applying results of Pitt and Valiant [30], assuming NP $\neq$ RP, we cannot PAC learn $\mathcal{C}_{k,n}$ if the hypothesis is constrained to come from $\mathcal{C}_{k',n}$ for any $k' \geq k$. Thus from Theorems 3 and 4, we cannot learn $\mathcal{C}_{k,n}$ using $\mathcal{C}_{k',n}$ in the on-line setting either. So it is necessary to use a more expressive hypothesis space. Thus to give even further evidence that the class of one-dimensional patterns is significantly more complex than the union of intervals on the real line, observe that the consistency problem for the latter class is trivial to solve.

Goldman and Scott [13] gave an efficient algorithm that uses the statistical query model [2, 18] to PAC learn the class of continuous one-dimensional geometric patterns under high noise rates (any rate $< 1/2$ of classification noise). They also performed an empirical study of how well their algorithm worked on both simulated and real data.

In Section 5 we give an on-line, agnostic algorithm for the concept class $\mathcal{C}_{k,n}$ when the widths of each target interval can vary. Then in Section 6 we generalize the above concept class to any constant dimension d. Specifically, we change the space from $\{1, \ldots, s\}$ to

$$\prod_{i=1}^{d} \{1, \ldots, s_i\}$$

where $s_i \in \{s_1, \ldots, s_d\}$ bounds the size of the space in dimension i. Now rather than representing each concept as a set of $\leq k$ intervals, we represent each concept as a set of $\leq k$ axis-parallel boxes, where the size of each box in each dimension can vary. In this class, classification of examples is similar to what is described above. Namely, an example is positive if and only if it satisfies the following two conditions.

1. Each of the n points in the instance is contained within one of the k axis-parallel boxes.
2. There is at least one of the n points in the instance contained within each of the k boxes.

Note that a d-dimensional version of the class defined in Definition 1 could represent its concepts as axis-parallel squares (all of the same size) if the L_∞ norm is used in the Hausdorff metric. Thus the class we study in Section 6 generalizes the class of geometric patterns under the L_∞ norm.

Finally, recall that our ultimate goal is to apply our learning algorithms to the landmark matching problem described in Section 2. Since the patterns derived from the signatures from the robot are discrete and can be bounded in all dimensions a priori, discretizing and bounding the input space is not a serious issue.

5 A General Learning Algorithm for Geometric Patterns

In this section we present the general framework we use to develop our agnostic algorithms. We first give a general definition for a class of geometric patterns and show how we apply Winnow. Then we use the virtual weights technique of Maass and Warmuth [28] to obtain an efficient agnostic algorithm.

5.1 Reduction to Winnow

In this section we describe how we convert the problem of learning geometric patterns into the problem of learning a disjunction over a set of variables, for which we can then apply Winnow. Let $\mathcal{C}$ be the concept class over the instance space $\mathcal{X}_n$ where each example in $\mathcal{X}_n$ is a collection of n points from the base domain $\mathcal{X}_{base}$. We represent each concept T in the touchstone class $\mathcal{T}$ by a set of at most k concepts from a base concept class $\mathcal{C}_{base}$ where the domain for $\mathcal{C}_{base}$ is $\mathcal{X}_{base}$. For example, when $\mathcal{T}$ is the class of one-dimensional geometric patterns, $\mathcal{X}_{base} = \{1, \ldots, s\}$ and $\mathcal{C}_{base}$ is the class of intervals over $\{1, \ldots, s\}$. Thus each concept $T \in \mathcal{T}$ is a collection of concepts from $\mathcal{C}_{base}$. Note that this class is a slight generalization of the discretized and bounded version of the class studied by Goldman and Scott [13] since in that paper, all intervals are the same fixed width, whereas here we allow the intervals' widths to vary.

The classification of an example $X \in \mathcal{X}_n$ with respect to a concept $T = \{t_1, ..., t_k\} \in \mathcal{T}$ is defined as follows. Each t_i is a concept from $\mathcal{C}_{base}$ that classifies a point p as positive if and only if t_i contains p. Example X is classified as positive by T if and only if the following two criteria are met.

Positive Criterion 1: For all j $(1 \leq j \leq k)$, there exists a point $p \in X$ for which $t_j(p) = 1$. That is, every concept in T classifies at least one point from X as positive.

Positive Criterion 2: For each point $p \in X$, there is some j $(1 \leq j \leq k)$ for which $t_j(p) = 1$. That is, every point in X is classified positively by some concept in T.

We assume the best hypothesis from $\mathcal{T}$ for the target concept is some collection of concepts $T = \{t_1, \ldots, t_k\}$ where each $t_j \in \mathcal{C}_{base}$. We now describe a key property used by our reduction. We require that we can represent the points in $\mathcal{X}_{base} \setminus (t_1 \cup \cdots \cup t_k)$ as a union of at most k_{comp} concepts from $\mathcal{C}_{base}$. That is, there must exist a set $T_{comp} = \{t'_1, \ldots, t'_{k_{comp}}\}$ where each $t'_j \in \mathcal{C}_{base}$ and $(t'_1 \cup \cdots \cup t'_{k_{comp}}) = \mathcal{X}_{base} \setminus (t_1 \cup \cdots \cup t_k)$. Furthermore, k_{comp} must be polynomial[8] in k. An efficient algorithm must run in time polynomial in $n \log |\mathcal{X}_{base}|$ (the number of bits to encode an example) and $k \log |\mathcal{C}_{base}|$ (the number of bits to define the target concept). In all of our applications, $\mathcal{C}_{base}$ is the class of d-dimensional axis-parallel boxes for d a constant. Since each box is defined by giving two points from $\mathcal{X}_{base}$, $|\mathcal{C}_{base}| = \Theta(|\mathcal{X}_{base}|^2)$. Thus $\log |\mathcal{C}_{base}| = \Theta(\log |\mathcal{X}_{base}|)$. Thus for an algorithm to be a polynomial time algorithm, it suffices for it to have complexity polynomial in n, k, and $\log |\mathcal{X}_{base}|$.

We now describe a transformation that reduces our learning problem to that of learning disjunctions of at most $2|\mathcal{C}_{base}|$ boolean attributes for which we can then apply Winnow. Note that an example $X \in \mathcal{X}_n$ violates Positive Criterion 2 if one of its points $p \in \mathcal{X}_{base}$ is classified positively by some concept in T_{comp} (which in turn means that p is classified as negative by all concepts in T). Similarly, an example $X \in \mathcal{X}_n$ violates Positive Criterion 1 if some concept in T classifies all points in X as negative. We now need to just introduce a set of boolean attributes to capture when either of the positive criteria is *violated*. We denote the set of attributes from which the concepts of T are selected by A, and the set of attributes

[8] We note that if each $t \in T$ can be represented by the intersection of a constant number of halfspaces, then k_{comp} is always polynomial in k.

from which the concepts of T_{comp} are selected by A_{comp}. We select the concepts for both T and T_{comp} from $\mathcal{C}_{base}$. Thus if computation time is not a concern, we can simply enumerate all such concepts. We associate each concept in $\mathcal{C}_{base}$ with two boolean attributes, placing one in A and one in A_{comp}. Given an example $X \in \mathcal{X}_n$ and an attribute $y \in A_{comp}$ we set $y = 1$ if and only if the concept from $\mathcal{C}_{base}$ associated with y classifies as positive at least one point p from X. As discussed above, Positive Criterion 2 is violated if and only if one of these attributes is set to 1. Similarly, given an example $X \in \mathcal{X}_n$ and an attribute $y \in A$ we set $y = 1$ if and only if the concept from $\mathcal{C}_{base}$ associated with y classifies *all* points p from X as negative. Positive Criterion 1 is violated if and only if one of these attributes is set to 1.

Let $\hat{T}$ be the disjunction of the attributes that correspond to the concepts in $T \cup T_{comp}$. Let $\hat{X}$ denote the assignment of values to the $2|\mathcal{C}_{base}|$ attributes obtained by the above transformation. Since $\hat{T}$ classifies example $\hat{X}$ as positive exactly when at least one of the positive criteria is violated (i.e. when X is negative), $\hat{T}(\hat{X}) = 1 - T(X)$. The number of transformed attributes is $2|\mathcal{C}_{base}|$ and the concepts in the touchstone class are disjunctions of at most $k + k_{comp}$ such attributes. If we run Winnow for this transformed instance space and touchstone class, by applying Theorems 1 and 2 (Section 3) with $N = 2|\mathcal{C}_{base}|$ and $K = k + k_{comp}$ we can guarantee the following mistake bounds.

Theorem 5 *Suppose the best (stationary) $T \in \mathcal{T}$ makes M_{opt} mistakes in a sequence of learning trials. Then the number of mistakes made by running Winnow on the transformed space is at most*

$$\begin{aligned} & (k + k_{comp})(2.75M_{opt} + 4.92(\ln(2|\mathcal{C}_{base}|) - 1)) + 4.92 \\ \leq\ & (k + k_{comp})(2.75M_{opt} + 4.92(\ln|\mathcal{C}_{base}| - 0.3)) + 4.92. \end{aligned}$$

If the concept is shifting (where Z is the number of shifts) and the best sequence of $T_t \in \mathcal{T}$ makes M_{opt} mistakes in a sequence of learning trials, then the number of mistakes made by running Winnow on the transformed space is at most

$$\begin{aligned} & 2.4(k + k_{comp})M_{opt} + 4.32Z(\ln(2|\mathcal{C}_{base}|) + 3.89) + \\ & 4.32\min\{2|\mathcal{C}_{base}|, Z\}(\ln(2|\mathcal{C}_{base}|) + 1.34) + 0.232 \\ \leq\ & 2.4(k + k_{comp})M_{opt} + 4.32Z(\ln|\mathcal{C}_{base}| + 4.59) + \\ & 4.32\min\{2|\mathcal{C}_{base}|, Z\}(\ln|\mathcal{C}_{base}| + 2.04) + 0.232. \end{aligned}$$

5.2 Efficient Implementation with Virtual Weights

The problem that remains with the direct implementation of Winnow over the $2|\mathcal{C}_{base}|$ attributes is that the number of attributes (and thus the computation time) is exponential. For example, when applied to the class of one-dimensional patterns where $\mathcal{C}_{base}$ is intervals over $\{1, \ldots, s\}$, $2|\mathcal{C}_{base}| = \Theta(s^2)$ which is exponential in $\log|\mathcal{X}_{base}| = \log|\{1, \ldots, s\}| = \log s$. We now use the virtual weight technique of Maass and Warmuth [28] to implicitly maintain the weights. The basic idea is to simulate Winnow by grouping concepts that "behave alike" into blocks. For each block only one weight has to be computed and we construct the blocks so that the number of concepts combined in each block as well as the weight for the block

can be efficiently computed. While the number of blocks increases as new counterexamples are received, the total number of blocks we create is polynomial in the number of mistakes. By applying the virtual weights technique, our time complexity will no longer depend polynomially on $|\mathcal{X}_{base}|$, but rather will depend polynomially in m, the total number of points from all examples the learner has misclassified. Note that m depends polynomially on the mistake bound[9] of our algorithm and n, the maximum number of points per example.

We group the weights associated with the attributes from A by using an adaptation of Maass and Warmuth's algorithm [28] for learning unions of boxes in fixed dimension. (The grouping of the weights for A_{comp} is similar.) The difference is that our examples are configurations of n points instead of single points. Suppose we want to predict the classification of an example X. Let $P = \{p_1, ..., p_m\}$ be the set of distinct points that appeared in the counterexamples (i.e. the examples that were misclassified). Note that m is at most n times the number of counterexamples seen so far.

We assume that $\mathcal{C}_{base}$ is a constant-dimensional box (which is the case for all our applications). Let d be the number of dimensions. Note that if each of the d axis-parallel $(d-1)$-dimensional hyperplanes is passed through each point in P, then at most $(m+1)^d$ regions are defined.

We now describe how we can group the attributes (i.e. boxes) from A such that we can efficiently compute how many of the attributes are in a group, and how much each box in a group contributes to the weighted sum used for prediction. Without loss of generality, assume the discrete values for dimension i are given by $\{1, \ldots, s_i\}$. For $1 \leq i \leq d$, let $\ell_{i,1}, \ldots, \ell_{i,m}$ be the dimension i coordinates of the points in P where $0 = \ell_{i,0} \leq \ell_{i,1} \leq \cdots \leq \ell_{i,m} \leq \ell_{i,m+1} = s_i$. Note that $\ell_{i,m+1}$ is set to s_i so that we can use the same notation to express the half interval $(\ell_{i,m}, s_i]$ as we do for the half-interval $(\ell_{i,j-1}, \ell_{i,j}]$ by setting $j = m+1$. Likewise, we want to be able to express the half-interval $(1, \ell_{i,1}]$ in this same manner. So that we do not need a special case when $\ell_{i,1} = 1$, we define $\ell_{i,0} = 0$.

Now for each of the at most $(m+1)^d$ vectors of the form $\vec{w} = (w_1, \ldots, w_d)$ where $1 \leq w_i \leq m+1$, we define the region

$$R_{\vec{w}} = (\ell_{1,w_1-1}, \ell_{1,w_1}] \times \cdots \times (\ell_{d,w_d-1}, \ell_{d,w_d}]$$

(see Figure 3). Let $\mathcal{R}$ be the collection of all such regions. As described in more depth below, we associate two groups with each region $R \in \mathcal{R}$ and 2^d groups with each pair of regions $R_{\vec{w}}, R_{\vec{z}} \in \mathcal{R}$ such that for all i, $\ell_{i,w_i} \leq \ell_{i,z_i}$. Thus together we have at most $O\left((m+1)^{2d}\right)$ groups to maintain (for constant d), which we can enumerate efficiently as long as m is polynomial in all relevant parameters. For a region $R_{\vec{w}}$ we define

$$|R_{\vec{w}}| = \prod_{i=1}^{d} (\ell_{i,w_i} - \ell_{i,w_i-1})$$

as the number of points that are in region $R_{\vec{w}}$. For each group G defined below, we describe how we efficiently count $|G|$, the number of attributes (i.e. boxes)[10] that are grouped into G.

[9]Note that for the shift-tolerant case, the mistake bound is not polynomial in $\log |\mathcal{C}_{base}|$ unless $Z = O\left(\log^c |\mathcal{C}_{base}|\right)$ for some constant c.

[10]Note that we include degenerate boxes (lines and points) in our counts. Simple modifications of our counting procedures will exclude degeneracies, if desired.

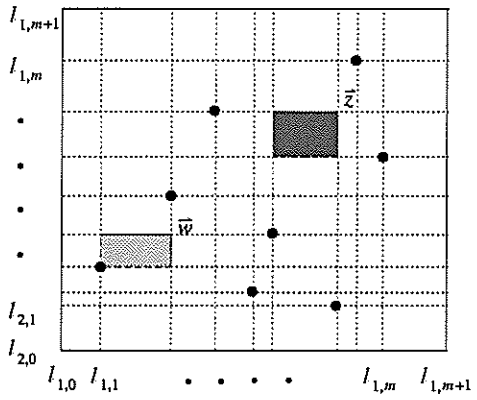

Figure 3: This demonstrates (for the case of $d = 2$) how we do the groupings for our virtual weight applications. The lightly shaded box (defined by $\vec{w}$ in its top right corner) is $R_{\vec{w}}$, and the more heavily shaded box (defined by $\vec{z}$ in its top right corner) is $R_{\vec{z}}$. The groups defined by $(R_{\vec{w}}, R_{\vec{z}})$ contain all boxes with the bottom left corner in the lightly shaded box and the top right corner in the heavily shaded box.

For the case when the boxes have both corners in some region, we define two groups, G_1 and G_2 for each such region $R_{\vec{w}}$. We must be careful since all boxes in a group must contain the same subset of points from P. Since the regions were formed by placing a hyperplane through each point from P and open on the "left side", the only point from P that could be in region is the "upper left" corner. Thus we have one group G_1 of boxes within $R_{\vec{w}}$ that contain the upper-left corner, and another group G_2 of boxes within $R_{\vec{w}}$ that do not contain the "upper left" corner. The sizes of these groups are

$$|G_1| = |R_{\vec{w}}| = \prod_{i=1}^{d} (\ell_{i,w_i} - \ell_{i,w_i-1})$$

and

$$|G_2| = \prod_{i=1}^{d} \left(\binom{\ell_{i,w_i} - \ell_{i,w_i-1}}{2} + (\ell_{i,w_i} - \ell_{i,w_i-1}) \right) - |G_1|.$$

We now consider the case in which the box is defined by two regions $R_{\vec{w}}$ and $R_{\vec{z}}$. Again, we must be careful since all boxes in a group must contain the same subset of points from P. As exemplified in Figure 3, the placement of the "lower left" corner of a box anywhere in $R_{\vec{w}}$ does not change which points in P that box contains. But such is not the case with the "upper right" corner in $R_{\vec{z}}$. Placing the "upper right" corner in any subset of the d outer hyperplanes of $R_{\vec{z}}$ (i.e. the one in each dimension that is farthest from $R_{\vec{w}}$) can cause the box to contain more points from P. Thus we use 2^d groups of boxes, each group with its "upper right" corner lying in a distinct subset of the outer hyperplanes of $R_{\vec{z}}$. Let

$$include_{\vec{z}}(j,i) = \begin{cases} 1 & \text{if bit } i \text{ is 1 in the binary representation of } j \\ (\ell_{i,z_i} - \ell_{i,z_i-1} - 1) & \text{otherwise} \end{cases}.$$

Then we can count number of boxes in group G_j for $j \in \left\{1, \ldots, 2^d\right\}$ as follows:

$$|G_j| = |R_{\vec{w}}| \cdot \prod_{i=1}^{d} include_{\vec{z}}(j, i).$$

Finally, we must compute the number of promotions and demotions (Section 3) that occur for each attribute in a group. It is easily verified that a previously seen counterexample sets the attributes corresponding to all boxes in a group to the same value. To count the number of promotions u (respectively, demotions v) that Winnow performs on the weight of each box in G, it is sufficient to go through the positive (respectively, negative) counterexamples and count how many of them set the attribute corresponding to any "token" box $b \in G$ to 1. If a new example X has a point in box b, then the contribution of *all* boxes in G to the weighted sum of the attributes that are turned on by X is simply $|G| \cdot \alpha^{u-v}$. If no point from X falls in the box b then all boxes in G make no contribution.

Theorem 6 *Geometric patterns defined over $\mathcal{X}_{base} = \{1, \ldots, s_1\} \times \cdots \times \{1, \ldots, s_d\}$ can be efficiently learned by Winnow using the touchstone class $\mathcal{T}$ of sets of at most k d-dimensional axis-parallel boxes. The mistake bound of Winnow with virtual weights on the transformed space is*

$$(k + k_{comp}) \left(2.75 M_{opt} + 4.92 \left(2 \left(\sum_{i=1}^{d} \ln s_i\right) - 0.3\right)\right) + 4.92$$

for the shift-free case, and

$$2.4\,(k + k_{comp})\, M_{opt} + 4.32 Z \left(4.59 + 2 \sum_{i=1}^{d} \ln s_i\right) +$$

$$4.32 \min\left\{2 \left(\prod_{i=1}^{d} s_i\right)^2, Z\right\} \left(2.04 + 2 \sum_{i=1}^{d} \ln s_i\right) + 0.232$$

for the shifting case. The time complexity is $O\left(m^{2d+1}\right)$ per trial. Here M_{opt} is the number of mistakes made by the best concept (or the best sequence of concepts) $T \in \mathcal{T}$, k_{comp} is the minimum number of d-dimensional axis-parallel boxes whose union comprises T's complement, m is at most n times the number of mistakes, and Z is the total number of shifts. The shift-free algorithm is always efficient and the shift-tolerant algorithm is efficient if $Z = O\left(\log^c\left(\prod_{i=1}^{d} s_i\right)\right)$ for some constant c.

Proof: We are simulating Winnow with $|\mathcal{C}_{base}| \leq |\mathcal{X}_{base}|^2 = \left(\prod_{i=1}^{d} s_i\right)^2$, so

$$\ln\left(2|\mathcal{C}_{base}|\right) \leq \ln\left(\sqrt{2}|\mathcal{X}_{base}|\right)^2 = 2\left(\ln\sqrt{2} + \sum_{i=1}^{d} \ln s_i\right) < 0.7 + 2\sum_{i=1}^{d} \ln s_i.$$

Applying Theorem 5 gives us the desired mistake bounds.

The number of groups is at most $O\left(m^{2d}\right)$ and to compute the weight of each attribute in each group takes $O(m)$ time. By inspection, the shift-free algorithm runs in time polynomial in $\log|\mathcal{X}_{base}|$, as does the shift-tolerant one if Z is dominated by a polynomial in $\log|\mathcal{X}_{base}|$. □

5.3 Application to Learning One-Dimensional Patterns

We now apply the above algorithm and corresponding results to the class of one-dimensional patterns as defined in Section 4 with the exception that we allow the target intervals to have arbitrary widths and we discretize and bound the space. Here $\mathcal{X}_{base} = \{1, \ldots, s\}$ (and thus each example consists of n points from $\{1, \ldots, s\}$), and $\mathcal{C}_{base}$ is the class of arbitrary width intervals over $\{1, \ldots, s\}$. The complement of the union of at most k intervals is the union of at most $k+1$ intervals. Hence $|\mathcal{C}_{base}| \leq s^2$ and $k_{comp} \leq k+1$, giving the following corollary of Theorem 6.

Corollary 7 *One-dimensional geometric patterns over the domain $\mathcal{X}_{base} = \{1, \ldots, s\}$ can be efficiently learned by Winnow using $\mathcal{T}$, the class of sets of at most k intervals. The mistake bound of Winnow with virtual weights on the transformed space is*

$$(2k+1)(2.75M_{opt} + 4.92(2\ln s - 0.3)) + 4.92$$

for the shift-free case, and

$$2.4(2k+1)M_{opt} + 4.32Z\,(4.59 + 2\ln s) + 4.32\min\left\{2s^2, Z\right\}(2.04 + 2\ln s) + 0.232$$

for the shifting case. The time complexity is $O(m^3)$ per trial. Here M_{opt} is the number of mistakes made by the best concept (or the best sequence of concepts) $T \in \mathcal{T}$, m is at most n times the number of mistakes, and Z is the total number of shifts. The shift-free algorithm is always efficient and the shift-tolerant algorithm is efficient if $Z = O(\log^c s)$ for some constant c.

6 Efficient Agnostic Learning of Two-Dimensional Patterns

To motivate our definition of a two-dimensional geometric pattern, we now review some findings from the experimental work of Goldman and Scott [13]. Their images (referred to as signatures) consisted of $s+1$ distinct light intensity values. (In the data from Pinette [29] that they used, $s = 359$.) Each signature is pre-processed by computing its first derivative and then normalizing it by dividing each of the s derivative values by the difference between the signature's maximum and minimum values. Let r denote the number of discrete values (precision) for these normalized derivative values. As r is increased more information is retained, but the complexity of the learning process (and thus the bounds on the number of prediction mistakes and the time needed to make a prediction) is increased. Next they obtained the training data for the class of one-dimensional patterns by converting the arrays of derivatives into one-dimensional geometric patterns by placing points where there were significant changes. Then they applied their algorithm, giving it a set of positive examples (patterns obtained from locations in the vicinity of the landmark) and a set of negative examples (patterns obtained from locations not in the vicinity of the landmark). Although

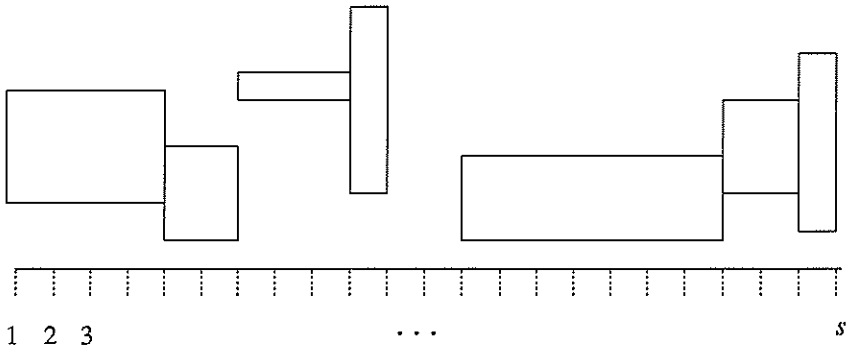

Figure 4: An example concept of a two-dimensional pattern as defined by non-overlapping rectangles. In general, we can also learn a concept of a constant-dimensional pattern defined by overlapping rectangles.

their experimental results were promising, too much important information was lost in moving from the signatures to the one-dimensional patterns. For example, the points do not reflect the magnitude of light intensity change, or even the direction of change (i.e. was the intensity increasing or decreasing). Since our algorithm is efficient for constant dimensional patterns, we can preserve more of the information from the original image. Of course there is a tradeoff between the number of dimensions and performance (as measured by both time complexity and learning performance).

In addition to the new algorithm we present and analyze, another contribution of this paper is a way to map the signatures to two-dimensional patterns in such a way that (1) all important information from the signatures is maintained and (2) we can successfully learn the resulting class of two-dimensional patterns. We now describe the class of two-dimensional patterns and how we propose the signatures be mapped to form the examples for our learning process. As we describe later, much more general classes of geometric patterns can be efficiently agnostically learned using our algorithm. We selected this particular class of patterns because we feel that it best fits the needs suggested by the experimental work of Goldman and Scott. However, learning more general classes of geometric patterns may also be useful in attacking the landmark matching problem. The process of mapping the original two-dimensional visual data to the signature involves extracting a small, contiguous set of rows from the eye level range of the two-dimensional data and averaging them into a one-dimensional signature. So while the mapping we describe retains all the important information from the one-dimensional signature, information is lost in going from the original two-dimensional visual data to the signature. Another important contribution of this paper is that our mapping and learning algorithm extend very naturally when working with the complete two-dimensional data by mapping it to a three (or higher)-dimensional pattern.

The touchstone class $\mathcal{T}$ we use is the class of sets of k non-overlapping[11] axis-parallel rectangles, where non-overlapping means that no two rectangles intersect when projected onto the x axis (see Figure 4). Note that each rectangle can have arbitrary y-values defining

[11]We later show that the restriction that the boxes be non-overlapping is not required for learning since we can learn when $\mathcal{T}$ consists of at most k arbitrary axis-parallel rectangles with a slight increase in complexity. For ease of exposition, we first consider the simpler case of non-overlapping rectangles.

its top and bottom—they need not "rest" on the x-axis.

The discrete values for the x-axis are $\{1, \ldots, s\}$, each value corresponding to a normalized derivative from the signature. Recall that we use r to denote the number of discrete values used for the normalized derivatives. Thus one can think of the rectangles being axis-parallel rectangles from the domain $\{1, \ldots, s\} \times \{1, \ldots, r\}$. Each example is an array of the s normalized derivatives obtained from a signature. The target concept can be thought of as a collection of allowable values (on the y axis) for each of the s normalized derivatives. Since these ranges of allowable values can vary within the same target concept, the concept can allow some values of the normalized derivative to vary more greatly than others. This is useful in cases where the signature comes from an environment where some objects are more reflective than others; reflective objects could produce higher variations of intensity in the signature than non-reflective objects. Also, since the target consists of variable-width rectangles, some of the derivative values can be "combined together" in a single rectangle in the target, allowing for examples (from real data sets) that undergo small translations of their points to still be correctly classified.

For learning the class of two-dimensional geometric patterns, our touchstone class consists of a collection of at most k non-overlapping axis-parallel boxes $T = \{t_1, \ldots, t_k\}$. An example X (a set of s points where the x-coordinate is an index into the array of derivatives, and the y-coordinate is the normalized value of the derivative) is classified as positive by T if and only if the following two criteria are met (where $1 \leq j \leq k$).

Positive Criterion 1: For all j, box $t_j \in T$ contains at least one point from X.

Positive Criterion 2: For each point $p \in X$, p is in box $t_j \in T$ for some j.

Note that here n, the maximum number of points in any example, is equal to s.

We now apply the algorithm and corresponding results from Section 5 to the situation in which the concept class $\mathcal{C}$ is the class of two-dimensional patterns. Here $\mathcal{X}_{base} = \{1, \ldots, s\} \times \{1, \ldots, r\}$ and $\mathcal{C}_{base}$ is the class of axis-parallel boxes from $\{1, \ldots, s\} \times \{1, \ldots, r\}$, so $|\mathcal{C}_{base}| \leq s^2 r^2$. $\mathcal{T}$ is the class of sets of at most k axis-parallel boxes from $\mathcal{C}_{base}$ with the restriction that no boxes in any $T \in \mathcal{T}$ overlap. Hence $k_{comp} \leq 2k + k + 1$ since for each box $t \in T \in \mathcal{T}$ at most two boxes are needed to cover the regions above and below t, and then at most one additional box is needed for each of the at most $k + 1$ regions between the boxes in T. Thus we obtain the following corollary of Theorem 6.

Corollary 8 *Two-dimensional geometric patterns from $\mathcal{X}_{base} = \{1, \ldots, s\} \times \{1, \ldots, r\}$ can be efficiently learned by Winnow using $\mathcal{T}$, the class of sets of at most k axis-parallel non-overlapping boxes. The mistake bound of Winnow with virtual weights on the transformed space is*

$$(4k + 1)\,(2.75 M_{opt} + 4.92\,(2 \ln s + 2 \ln r - 0.3)) + 4.92$$

for the shift-free case, and

$$2.4(4k + 1) M_{opt} + 4.32 Z\,(4.59 + 2 \ln s + 2 \ln r) +$$
$$4.32 \min\left\{2s^2 r^2, Z\right\}(2.04 + 2 \ln s + 2 \ln r) + 0.232$$

for the shifting case. The time complexity is $O(m^5)$ per trial. Here M_{opt} is the number of mistakes made by the best concept (or the best sequence of concepts) $T \in \mathcal{T}$, m is at most n times the number of mistakes, and Z is the total number of shifts. The shift-free algorithm is always efficient and the shift-tolerant algorithm is efficient if $Z = O(\log^c(rs))$ for some constant c.

One might ask why we chose a new learning model and a different approach to develop a d-dimensional pattern-learning algorithm rather than extending the algorithm of Goldman and Scott to higher dimensions. One reason is simplicity: the algorithm we present here is substantially easier to develop, analyze, and understand than that of Goldman and Scott. A second reason is that this paper's algorithm was developed in the on-line learning model, whereas Goldman and Scott's algorithm only runs in batch mode. It is well known that efficient on-line algorithms can be easily turned into efficient PAC algorithms (Section 3), but not all efficient PAC algorithms have efficient on-line counterparts [1, 24]. Thus a robot equipped with this paper's algorithm can learn as it goes, but one using Goldman and Scott's algorithm cannot.

Another advantage our new algorithm has over the one of Goldman and Scott is that it is robust against attribute errors, which can be interpreted in many ways (Section 3): tolerance of noise in the attributes, tolerance of classification noise, or as an agnostic algorithm. This last interpretation means that we can think of the target concept as *any* classifier (even one not in the hypothesis class) and our algorithm will attempt to find the best hypothesis from its hypothesis class, making no assumptions whatsoever about the target. Even in those cases, we can make guarantees about the algorithm's performance with respect to the best it could do if it knew which hypothesis in its class would have had the lowest error. In contrast, the performance guarantees for the algorithm of Goldman and Scott break down if the assumptions about the concept class and noise model are not met.

Finally, the algorithm we present here can track a shifting concept. This is done without any knowledge about when or how much the target concept changes. In contrast, the algorithm of Goldman and Scott does not tolerate concept shift. In fact, the only way to track a changing concept with that algorithm is to abandon the old hypothesis when the target concept changes, acquire a new training set, and rerun the algorithm. But this requires knowledge of exactly when the concept changed.

7 Further Extensions

We chose the extension discussed in the previous section because it seemed to fit the application of landmark matching from one-dimensional data. However, there are other more general classes that our algorithm can be easily adapted to learn. For example, we can extend our touchstone class to be defined by up to k arbitrary (possibly intersecting) axis-parallel d-dimensional boxes where d is any constant. For ease of exposition, we assume that $\mathcal{X}_{base} = \{1, \ldots, s\}^d$. Thus $\mathcal{C}_{base}$ is the class of axis-parallel boxes from $\{1, \ldots, s\}^d$, so $|\mathcal{C}_{base}| \leq s^{2d}$. $\mathcal{T}$ is the class of sets of at most k arbitrary axis-parallel d-dimensional boxes. To upperbound k_{comp}, note that projecting the boxes onto each of the d axes yields $\leq 2k+1$ segments on each axis. These segments define all the possible gaps that must be filled by

boxes to yield T_{comp}, so the total number of gaps is $\leq (2k+1)^d$. Each gap can be filled with a single box, so $k_{comp} \leq (2k+1)^d$. Thus we obtain the following corollary of Theorem 6.

Corollary 9 *The class of d-dimensional geometric patterns from $\mathcal{X}_{base} = \{1, \ldots, s\}^d$ can be efficiently learned by Winnow using $\mathcal{T}$, the class of sets of at most k arbitrary axis-parallel d-dimensional boxes. The mistake bound of Winnow with virtual weights on the transformed space is*

$$\left(k + (2k+1)^d\right)\left(2.75M_{opt} + 4.92\left(2d\ln s - 0.3\right)\right) + 4.92$$

for the shift-free case, and

$$2.4\left(k + (2k+1)^d\right)M_{opt} + 4.32Z\left(4.59 + 2d\ln s\right) + 4.32\min\left\{2s^{2d}, Z\right\}\left(2.04 + 2d\ln s\right) + 0.232$$

for the shifting case. The time complexity is $O\left(m^{2d+1}\right)$ per trial. Here M_{opt} is the number of mistakes made by the best concept (or the best sequence of concepts) $T \in \mathcal{T}$, m is at most n times the number of mistakes, and Z is the total number of shifts. The shift-free algorithm is always efficient and the shift-tolerant algorithm is efficient if $Z = O\left((d\log s)^c\right)$ for some constant c.

Just as we converted the one-dimensional data into a two-dimensional pattern by using the light intensities as the second dimension, using this same method two-dimensional data can be converted into a three-dimensional pattern. If the target boxes obtained are non-overlapping then better bounds than those from Corollary 9 can be obtained.

Finally, Corollary 9 generalizes an algorithm to agnostically learn with a touchstone class of discrete two-dimensional patterns under a variation of the Hausdorff metric (Section 4) in which the L_∞ versus the L_2 norm is used. The continuous version using the L_2 norm was studied by Goldberg [12], yielding a PAC algorithm. While our class is more restrictive, our algorithm is on-line, agnostic, and can handle concept shift.

8 Learning from Multiple-Instance Examples

Recently, motivated by drug discovery, Dietterich et al. [10] introduced the notion of learning from multiple-instance examples where the target concept is simply a boolean function, each example is a collection of instances, and the example (collection) is classified as positive if and only if at least one of its elements is mapped to 1 by the target concept. Their model is primarily motivated by the problem of predicting whether a molecule would bind at a particular site. They argued empirically that axis-parallel rectangles are good hypotheses for this and other similar learning problems. Subsequently, Long and Tan [27] described an efficient PAC algorithm for learning a single axis-parallel box in $\mathbf{Q}^d$ (where $\mathbf{Q}$ denotes the set of rationals) from multiple-instance examples if each instance is drawn independently from a product distribution and d need not be constant. Auer et al. [4] gave an efficient PAC algorithm for learning a single axis-parallel box in $\Re^d$ from multiple-instance examples if each instance is drawn independently from an arbitrary distribution over $\Re^d$. This algorithm is also polynomial in d. Later, Auer [3] modified that algorithm (making it more practical) and

gave an empirical analysis of the modified algorithm. In the above papers, each example is classified as positive if at least one of its points is inside the target box.

Our algorithm for learning constant-dimensional patterns learns a *union* of axis-parallel boxes from a constant-dimensional, finite, discretized space where a multiple-instance example is classified as positive if and only if (1) each point is classified as positive by some box and (2) every box contains at least one point. Furthermore, the algorithm of Corollary 9 is easily adapted to agnostically learn the union of axis-parallel boxes (of constant dimension) in the multiple-instance model under the rule that an example is positive if at least one of its points is inside some target box. We achieve this result by setting the attributes of A according to the rule for the attributes of A_{comp} and ignoring the attributes of A_{comp} (where A and A_{comp} are defined as in Section 5.1). In addition, other variations for the rule of when an example is positive can be used.

9 Concluding Remarks

In this paper we presented a new approach to learning geometric patterns. By discretizing and bounding the space (which causes no problems for our intended application area), we are able to learn a generalization of d-dimensional patterns (for fixed d) under the L_∞ norm in an on-line setting with an agnostic, shift-tolerant algorithm. As mentioned in Section 3, the mistake bounds of our algorithm can be improved if we tune Winnow using knowledge of k, k_{comp}, the number of shifts, and the number of attribute errors. One method of tuning Winnow is to apply the weighted majority algorithm, as demonstrated by Littlestone and Warmuth [26].

Note that we need not require the coordinates in each dimension to be integers from $\{1, \ldots, s\}$, so long as there are a finite number of them. In fact, the density of the coordinates along an axis could vary, e.g. $\{1, 5/4, 3/2, 7/4, 2, 3, 5\}$. We would require our algorithm to have time complexity and a mistake bound that are polynomial in the size of the specification of the coordinates. For example, the coordinates in each dimension could be specified by a sequence of triples $\langle (a_i, b_i, t_i) \rangle$, $1 \leq i \leq w$, where $[a_i, b_i] \subseteq [1, s]$ represents an interval on the current axis and t_i is the number of points uniformly distributed in $[a_i, b_i]$ (we would require $[a_i, b_i] \cap [a_j, b_j] = \emptyset$ for all $i \neq j$). In this case, our algorithm's time complexity and mistake bound must be polynomial in $O\left(w \left(\log \left(\max_i t_i\right) + \log s\right)\right)$, which is polynomial in $\log s$ if there exist constants c_1 and c_2 such that $t_i = O\left(s^{c_1}\right)$ for all i and $w = O\left(\log^{c_2} s\right)$. If this is the input representation used, then we can still apply the virtual weights technique in time polynomial in the input size. To efficiently compute the number of boxes in a group, we merely need to efficiently count the number of coordinates in each dimension between any two points. This can be easily done since we can check all triples in $O(w)$ time and we can count the number of coordinates to the left or right of any point in any triple: given a point $p \in [a_i, b_i]$, the number of coordinates in $[a_i, b_i]$ strictly to the left of p is $(p - a_i)(t_i - 1)$. The coordinates to the right of p are counted in a similar fashion.

Recall from Section 3 that we can convert our on-line algorithms into PAC algorithms. If the mistake bound is known a priori, we can apply Theorem 4. Otherwise we can apply Theorem 3, which costs us a factor of M in the sample complexity. It should also be possible to remove the restriction of the discretized and bounded space when converting this algorithm

into a PAC algorithm. The learner would draw a sufficiently large (polynomially sized) set of examples and then use these examples to build a polynomially sized set of candidate boxes, a subset of which will comprise the hypothesis.

Removing the exponential dependence on d from our time bounds would be a very difficult task. It is well known (see e.g. Maass and Warmuth [28] or Bshouty et al. [7]) that learning unions of at most k d-dimensional boxes (with single-instance examples) in time polynomial in d with $\mathcal{X}_{base} = \{0,1\}^d$ yields an algorithm for learning k-term DNF formulas over d variables in time polynomial in k and d. Each term in the DNF maps to a box, and each single-instance example (assignment to the d variables) maps to a point in $\mathcal{X}_{base}$. Since this problem is a major open one in computational learning theory, it is unlikely that a simple solution exists. Because our algorithm generalizes learning unions of axis-parallel boxes, removing the exponential dependence on d would be a major accomplishment.

One future direction is to perform experimental work on Pinette's signatures, comparing our empirical results to those of Goldman and Scott. We can also run the algorithms on real data from other domains, perhaps including data represented as two-dimensional arrays (e.g. two-dimensional images). Additionally, a potential advantage of the approach of this paper is that our hypothesis can be very naturally used to associate a confidence measure to a prediction that an example is negative. Recall that a hypothesis produced by our algorithm is a disjunction of attributes that describe when the example is negative. Thus if several attributes in the hypothesis are satisfied, then we have a high confidence that the current location is not near the landmark. However, if a single attribute in the hypothesis is satisfied, then the learner has a much lower confidence about its prediction that the current location is not near a landmark. This additional information could be valuable for a navigation system. Along this direction, it would be interesting to modify the algorithm so that it made a real-valued prediction giving a measure of confidence that there is a match with the landmark.

Another avenue of future work is converting these algorithms from landmark *matchers* to landmark *recognizers*, i.e. a system that can determine which landmark it thinks it is closest to. To do this, our algorithms need to learn *multi-valued* (versus only binary) functions. One approach to converting binary function learners to multi-valued function learners is via the use of error-correcting output codes (ECOCs) [32, 9]. Here the learner learns a collection of classifiers, each classifier providing one bit of the code for a particular value. ECOCs are used so that multiple codes map to the same value, making the learning algorithm more robust. These methods have a successful empirical history [6, 8, 9, 36, 21].

Finally, another interesting direction is to further study the possibilities for applying our techniques to learn other geometric concepts under the multiple-instance model as well as exploring other meaningful rules to classify the multiple-instance examples.

Acknowledgments

We thank Manfred Warmuth and Subhash Suri for valuable discussions about this work. We also thank the COLT committee members for their comments on an earlier draft of this paper. Stephen Scott and Stephen Kwek performed this work at Washington University.

References

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[2] J. A. Aslam and S. E. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Journal of Computer and System Sciences*, 1998. To appear.

[3] P. Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*, pages 21–29. Morgan Kaufmann, 1997.

[4] P. Auer, P. M. Long, and A. Srinivasan. Approximating hyper-rectangles: Learning and pseudo-random sets. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 314–323. ACM, 1997.

[5] Peter Auer and Manfred Warmuth. Tracking the best disjunction. *Machine Learning*, 1998. To appear in the Special Issue on Context Sensitivity and Concept drift.

[6] G. Bakiri. Converting English text to speech: A machine learning approach. Technical Report 91-30-2, Oregon State University, Corvallis, OR, 1991.

[7] N.H. Bshouty, P.W. Goldberg, S.A. Goldman, and H.D. Mathias. Exact learning of discretized geometric concepts. *SIAM Journal of Computing*, 1998. To appear.

[8] T. G. Dietterich and G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of AAAI '91*, pages 572–577. AAAI Press/MIT Press, 1991.

[9] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

[10] Thomas G. Dietterich, Richard H. Lathrop, and Tomas Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 1996. To appear.

[11] P. Goldberg, S. A. Goldman, and S. D. Scott. PAC learning of one-dimensional patterns. *Machine Learning*, 25(1):51–70, October 1996.

[12] Paul Goldberg. *PAC Learning Geometrical Figures.* PhD thesis, University of Edinburgh, 1992.

[13] S. A. Goldman and S. D. Scott. A theoretical and empirical study of a noise-tolerant algorithm to learn geometric patterns. *Machine Learning*, 1998. To appear.

[14] P. M. Gruber. Approximation of convex bodies. In P. M. Gruber and J. M. Willis, editors, *Convexity and its Applications.* Brikhäuser Verlag, 1983.

[15] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inform. Comput.*, 100(1):78–150, September 1992.

[16] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. *Inform. Comput.*, 95(2):129–161, December 1991.

[17] J. Hong, X. Tan, B. Pinette, R. Weiss, and E. Riseman. Image-based homing. *IEEE Control Systems Magazine*, 12(1):38–45, 1992.

[18] M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 392–401. ACM Press, New York, NY, 1993.

[19] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2/3):115–142, 1994.

[20] J. Kivinen, M. K. Warmuth, and P. Auer. The perceptron algorithm vs. Winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 1998. To appear.

[21] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proc. 12th International Conference on Machine Learning*, pages 313–321. Morgan Kaufmann, 1995.

[22] T. Levitt and D. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(3):305–360, 1990.

[23] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[24] N. Littlestone. From on-line to batch learning. In *Proc. 2nd Annu. Workshop on Comput. Learning Theory*, pages 269–284, San Mateo, CA, 1989. Morgan Kaufmann.

[25] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 147–156, San Mateo, CA, 1991. Morgan Kaufmann.

[26] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[27] Philip M. Long and Lei Tan. PAC learning axis-aligned rectangles with respect to product distributions from multiple-instance examples. In *Proc. 9th Annu. Conf. on Comput. Learning Theory*, pages 228–234. ACM Press, New York, NY, 1996.

[28] Wolfgang Maass and Manfred K. Warmuth. Efficient learning with virtual threshold gates. *Information and Computation*, 1998. To appear.

[29] B. Pinette. *Image-Based Navigation Through Large-Scaled Environments.* PhD thesis, University of Massachusetts, Amherst, 1993.

[30] L. Pitt and L. Valiant. Computational limitations on learning from examples. *J. ACM*, 35:965–984, 1988.

[31] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[32] R. E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*, pages 313–321. Morgan Kaufmann, 1997.

[33] H. Suzuki and S. Arimoto. Visual control of autonomous mobile robot based on self-organizing model for pattern learning. *Journal of Robotic Systems*, 5(5):453–470, 1988.

[34] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[35] L. G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, vol. 1*, pages 560–566, Los Angeles, California, 1985. International Joint Committee for Artificial Intelligence.

[36] D. Wettschereck and T. G. Dietterich. Improving the performance of radial basis function networks by learning center locations. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems, 4*, pages 1133–1140. Morgan Kaufmann, San Francisco, CA, 1992.